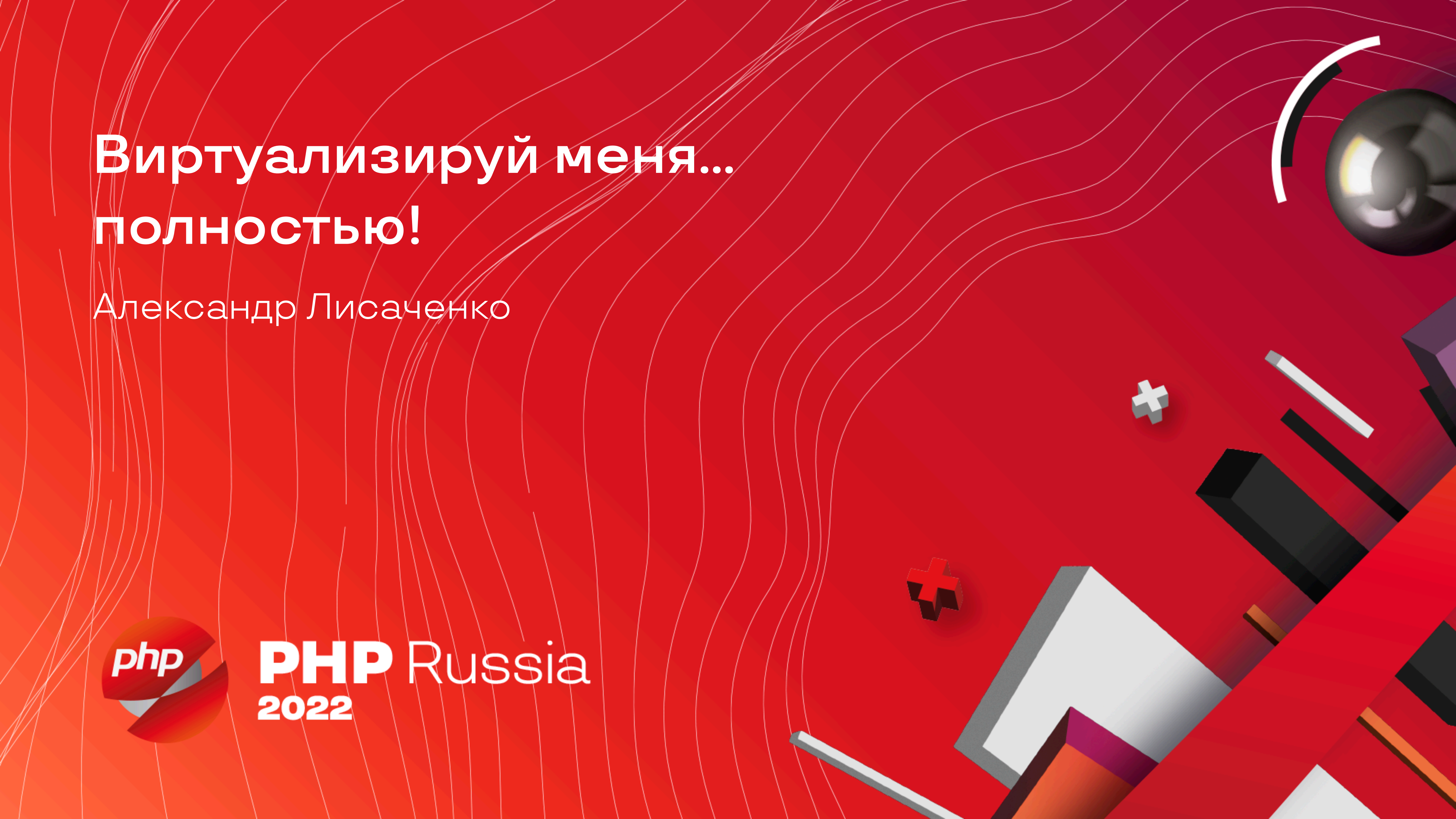

Виртуализируй меня…
полностью!
Александр Лисаченко
php
PHP Russia
2022

- Software Architect at inDrive
- Have worked with computers since 7 years
- Clean code advocate, guru in the Enterprise Architecture
- Author of the aspect-oriented framework Go! AOP http://go.aopphp.com and Z-engine https://github.com/lisachenko/z-engine





Agenda

- History of PHP environment evolution
    - Running PHP on local environment
    - Running PHP in Docker environments
    - Tips and tricks for running PHP in multi-environment modes with «docker compose»
    - Running PHP in Kubernetes environments



History of PHP
environment evolution
museum.php.net



Index of /

File name	File size	Date
patches/	-	2014-Nov-05 20:43
php-gtk/	-	2014-Nov-05 20:43
php1/	-	2014-Nov-05 20:43
php2/	-	2014-Nov-05 20:43
php3/	-	2014-Nov-05 20:43
php4/	-	2014-Nov-05 20:45
php5/	-	2020-Jul-07 10:32
php7/	-	2021-Nov-25 13:19
php8/	-	2021-Nov-25 13:27
win32/	-	2014-Nov-05 20:50



Building from scratch 😋

```
/work
❯ sudo apt install build-essential autoconf libtool bison re2c
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  automake autotools-dev g++ g++-10 libltdl-dev libstdc++-10-dev m4
Suggested packages:
  autoconf-archive gnu-standards autoconf-doc gettext bison-doc g++-multilib g++-10-multilib
  gcc-10-doc libtool-doc libstdc++-10-doc gfortran | fortran95-compiler gcj-jdk m4-doc
The following NEW packages will be installed:
  autoconf automake autotools-dev bison build-essential g++ g++-10 libltdl-dev libstdc++-10-dev
  libtool m4 re2c
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 14.0 MB of archives.
After this operation, 57.7 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```



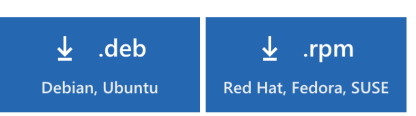
Pre-built PHP packages via package-managers





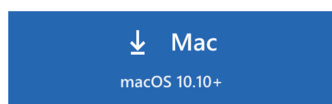
User Installer 64 bit 32 bit ARM
System Installer 64 bit 32 bit ARM
.zip 64 bit 32 bit ARM





.deb 64 bit ARM ARM 64
.rpm 64 bit ARM ARM 64
.tar.gz 64 bit ARM ARM 64
Snap Store



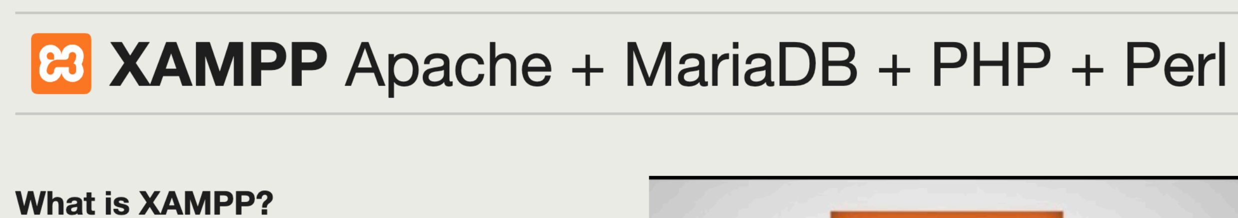


.zip Universal Intel Chip Apple Silicon



Old classic XAMPP

apachefriends.org



XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.



Download
Click here for other versions

XAMPP for **Windows** 8.1.10 (PHP 8.1.10)

XAMPP for **Linux** 8.1.10 (PHP 8.1.10)

XAMPP for **OS X** 8.1.6 (PHP 8.1.6)



Running PHP in the Docker



WHEN YOU TRY TO EXPLAIN

WHAT DOCKER IS

meme-arsenal.ru



Php one-line alias

```
❯ alias php='docker run -it --rm --name php-cli -p8080:8080 -v
"$PWD":"$PWD" -w "$PWD" php:8.2.0RC6-cli-alpine'

❯ php -v
PHP 8.2.0RC6 (cli) (built: Nov 15 2022 04:15:32) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.2.0RC6, Copyright (c) Zend Technologies


❯ php -S 0.0.0.0:8080 -t ./
[Tue Nov 15 20:51:39 2022] PHP 8.2.0RC6 Development Server (http://
0.0.0.0:8080) started
```



Php one-line alias

```
❯ alias php='docker run -it --rm --name php-cli -p8080:8080 -v
"$PWD":"$PWD" -w "$PWD" php:8.2.0RC6-cli-alpine'
```

```
❯ php -v
PHP 8.2.0RC6 (cli) (built: Nov 15 2022 04:15:32) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.2.0RC6, Copyright (c) Zend Technologies
```

```
❯ php -S 0.0.0.0:8080 -t ./
[Tue Nov 15 20:51:39 2022] PHP 8.2.0RC6 Development Server (http://
0.0.0.0:8080) started
```

Php one-line alias

```
❯ alias php='docker run -it --rm --name php-cli -p8080:8080 -v
"$PWD":"$PWD" -w "$PWD" php:8.2.0RC6-cli-alpine'
```

```
❯ php -v
PHP 8.2.0RC6 (cli) (built: Nov 15 2022 04:15:32) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.2.0RC6, Copyright (c) Zend Technologies
```

```
❯ php -S 0.0.0.0:8080 -t ./
[Tue Nov 15 20:51:39 2022] PHP 8.2.0RC6 Development Server (http://
0.0.0.0:8080) started
```



Php one-line alias

```
❯ alias php='docker run -it --rm --name php-cli -p8080:8080 -v
"$PWD":"$PWD" -w "$PWD" php:8.2.0RC6-cli-alpine'

❯ php -v
PHP 8.2.0RC6 (cli) (built: Nov 15 2022 04:15:32) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.2.0RC6, Copyright (c) Zend Technologies
```

```
❯ php -S 0.0.0.0:8080 -t ./
[Tue Nov 15 20:51:39 2022] PHP 8.2.0RC6 Development Server (http://
0.0.0.0:8080) started
```



Custom PHP images

```
FROM php:8.2.0RC6-cli
RUN apt-get update && apt-get install -y \
      libfreetype6-dev \
      libjpeg62-turbo-dev \
      libpng-dev \
   && docker-php-ext-configure gd --with-freetype --with-jpeg \
   && docker-php-ext-install -j$(nproc) gd
```



Custom PHP images

```
FROM php:8.2.0RC6-cli
RUN apt-get update && apt-get install -y \
      libfreetype6-dev \
      libjpeg62-turbo-dev \
      libpng-dev \
   && docker-php-ext-configure gd --with-freetype --with-jpeg \
   && docker-php-ext-install -j$(nproc) gd
```



Custom PHP images

```
FROM php:8.2.0RC6-cli
RUN apt-get update && apt-get install -y \
        libfreetype6-dev \
        libjpeg62-turbo-dev \
        libpng-dev \
    && docker-php-ext-configure gd --with-freetype --with-jpeg \
    && docker-php-ext-install -j$(nproc) gd
```



Php extension dependencies

```
libfreetype6-dev \
libjpeg62-turbo-dev \
libpng-dev
```



Php extension dependencies

```
libfreetype6-dev \
libjpeg62-turbo-dev \
libpng-dev
```



Php extension dependencies

```
libfreetype6-dev \
libjpeg62-turbo-dev \
libpng-dev
```





Php extension installer

https://github.com/mlocati/docker-php-extension-installer



Php extension installer

https://github.com/mlocati/docker-php-extension-installer

Easy installation of PHP extensions in official PHP Docker images

This repository contains a script that can be used to easily install a PHP extension inside the official PHP Docker images.

The script will install all the required APT/APK packages; at the end of the script execution, the no-more needed packages will be removed so that the image will be much smaller.



Php extension installer

https://github.com/mlocati/docker-php-extension-installer

Easy installation of PHP extensions in official PHP Docker images

This repository contains a script that can be used to easily install a PHP extension inside the official PHP Docker images.

The script will install all the required APT/APK packages; at the end of the script execution, the no-more needed packages will be removed so that the image will be much smaller.



Custom PHP images - Before

```
FROM php:8.2.0RC6-cli-alpine
RUN apt-get update && apt-get install -y \
      libfreetype6-dev \
      libjpeg62-turbo-dev \
      libpng-dev \
   && docker-php-ext-configure gd --with-freetype --with-jpeg \
   && docker-php-ext-install -j$(nproc) gd
```



Custom PHP images - After

```
FROM php:8.2.0RC6-cli-alpine
COPY --from=mlocati/php-extension-installer \
     /usr/bin/install-php-extensions /usr/local/bin/

RUN install-php-extensions gd
```



Custom PHP images - After

```
FROM php:8.2.0RC6-cli-alpine
COPY --from=mlocati/php-extension-installer \
     /usr/bin/install-php-extensions /usr/local/bin/

RUN install-php-extensions gd
```



Custom PHP images - After

```
FROM php:8.2.0RC6-cli-alpine
COPY --from=mlocati/php-extension-installer \
     /usr/bin/install-php-extensions /usr/local/bin/

RUN install-php-extensions gd
```







docker build

docker buildx build



Dealing with secrets and SSH - Before

```
FROM php:8.2.0RC6-cli-alpine
RUN apk add --no-cache openssh-client \
# Build-time variable passing
ARG SSH_PRIVATE_KEY
RUN mkdir /root/.ssh/
RUN echo "${SSH_PRIVATE_KEY}" > /root/.ssh/id_rsa

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts
```



Dealing with secrets and SSH - Before

```
FROM php:8.2.0RC6-cli-alpine
RUN apk add --no-cache openssh-client \
# Build-time variable passing
ARG SSH_PRIVATE_KEY
RUN mkdir /root/.ssh/
RUN echo "${SSH_PRIVATE_KEY}" > /root/.ssh/id_rsa

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts
```



Dealing with secrets and SSH - Before

```
FROM php:8.2.0RC6-cli-alpine
RUN apk add --no-cache openssh-client \
# Build-time variable passing
ARG SSH_PRIVATE_KEY
RUN mkdir /root/.ssh/
RUN echo "${SSH_PRIVATE_KEY}" > /root/.ssh/id_rsa

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts
```

no, thanks



Dealing with secrets and SSH - Better

```
FROM php:8.2.0RC6-cli-alpine as build
RUN apk add --no-cache openssh-client
# Build-time variable passing
ARG SSH_PRIVATE_KEY
RUN mkdir /root/.ssh/
RUN echo "${SSH_PRIVATE_KEY}" > /root/.ssh/id_rsa

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts

# Isolated stage without secrets
FROM php:8.2.0RC6-cli-alpine
COPY --from=build /build /app
```



Dealing with secrets and SSH - Better

```
FROM php:8.2.0RC6-cli-alpine as build
RUN apk add --no-cache openssh-client
# Build-time variable passing
ARG SSH_PRIVATE_KEY
RUN mkdir /root/.ssh/
RUN echo "${SSH_PRIVATE_KEY}" > /root/.ssh/id_rsa

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts

# Isolated stage without secrets
FROM php:8.2.0RC6-cli-alpine
COPY --from=build /build /app
```



Dealing with secrets and SSH - Better

```
FROM php:8.2.0RC6-cli-alpine as build
RUN apk add --no-cache openssh-client
# Build-time variable passing
ARG SSH_PRIVATE_KEY
RUN mkdir /root/.ssh/
RUN echo "${SSH_PRIVATE_KEY}" > /root/.ssh/id_rsa

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts

# Isolated stage without secrets
FROM php:8.2.0RC6-cli-alpine
COPY --from=build /build /app
```

PHP Russia 2022

Dealing with secrets and SSH - After

```
# syntax=docker/dockerfile:1
FROM php:8.2.0RC6-cli-alpine
RUN apk add --no-cache openssh-client

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts

# BuildKit SSH type mount \
RUN --mount=type=ssh \
  ssh -q -T git@bitbucket.org 2>&1
```



Dealing with secrets and SSH - After

```
# syntax=docker/dockerfile:1
FROM php:8.2.0RC6-cli-alpine
RUN apk add --no-cache openssh-client

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts

# BuildKit SSH type mount \
RUN --mount=type=ssh \
  ssh -q -T git@bitbucket.org 2>&1
```



Dealing with secrets and SSH - After

```
# syntax=docker/dockerfile:1
FROM php:8.2.0RC6-cli-alpine
RUN apk add --no-cache openssh-client

# Add domain to known hosts
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts

# BuildKit SSH type mount \
RUN --mount=type=ssh \
  ssh -q -T git@bitbucket.org 2>&1
```



Dealing with secrets and SSH - After

```
eval $(ssh-agent)
ssh-add ~/.ssh/id_rsa
(Input your passphrase here)

docker build --ssh default=$SSH_AUTH_SOCK .
```



Dealing with secrets and SSH - After

```
eval $(ssh-agent)
ssh-add ~/.ssh/id_rsa
(Input your passphrase here)
```

```
docker build --ssh default=$SSH_AUTH_SOCK .
```



Dealing with Composer secrets- After

```
# syntax=docker/dockerfile:1
FROM dev as builder

ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=cache,id=cache,target=/tmp/composer/cache \
    --mount=type=secret,id=auth.json,target=/tmp/composer/auth.json,required \
    --mount=type=ssh \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



Dealing with Composer secrets- After

```
# syntax=docker/dockerfile:1
FROM dev as builder

ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=cache,id=cache,target=/tmp/composer/cache \
    --mount=type=secret,id=auth.json,target=/tmp/composer/auth.json,required \
    --mount=type=ssh \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



Dealing with Composer secrets- After

```
# syntax=docker/dockerfile:1
FROM dev as builder

ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=cache,id=cache,target=/tmp/composer/cache \
    --mount=type=secret,id=auth.json,target=/tmp/composer/auth.json,required \
    --mount=type=ssh \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



Dealing with Composer secrets- After

```
# syntax=docker/dockerfile:1
FROM dev as builder

ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=cache,id=cache,target=/tmp/composer/cache \
    --mount=type=secret,id=auth.json,target=/tmp/composer/auth.json,required \
    --mount=type=ssh \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



Dealing with Composer secrets- After

```
# syntax=docker/dockerfile:1
FROM dev as builder

ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=cache,id=cache,target=/tmp/composer/cache \
    --mount=type=secret,id=auth.json,target=/tmp/composer/auth.json,required \
    --mount=type=ssh \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



Dealing with Composer secrets- After

```
# syntax=docker/dockerfile:1
FROM dev as builder

ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=cache,id=cache,target=/tmp/composer/cache \
    --mount=type=secret,id=auth.json,target=/tmp/composer/auth.json,required \
    --mount=type=ssh \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



Dealing with Composer secrets- After

```
# syntax=docker/dockerfile:1
FROM dev as builder

ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=cache,id=cache,target=/tmp/composer/cache \
    --mount=type=secret,id=auth.json,target=/tmp/composer/auth.json,required \
    --mount=type=ssh \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



Dealing with Composer secrets- After

```
# syntax=docker/dockerfile:1
FROM dev as builder

ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=cache,id=cache,target=/tmp/composer/cache \
    --mount=type=secret,id=auth.json,target=/tmp/composer/auth.json,required \
    --mount=type=ssh \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```




BUILDING IMAGES
FROM SCRATCH


BUILDING
FROM BASE IMAGE


php
PHP Russia
2022

Base PHP image per company

```
# syntax=docker/dockerfile:1
# Configure version via --build-arg PHP_IMAGE=8.1-cli This image is for alpine only.

ARG PHP_IMAGE=8
FROM php:${PHP_IMAGE}-alpine as builder

COPY --from=mlocati/php-extension-installer /usr/bin/install-php-extensions /usr/bin/

RUN <<EOF
apk --update add --no-cache --virtual .build-deps $PHPIZE_DEPS
apk --update add --no-cache libressl bash unzip
install-php-extensions apcu intl redis opcache pcov pdo_mysql xdebug zip
apk del .build-deps
rm -rf /usr/local/etc/php/conf.d/docker-php-ext-{pcov,xdebug}.ini
EOF

COPY --link rootfs /

# See https://getcomposer.org/doc/03-cli.md#composer-home
ENV COMPOSER_HOME=/tmp/composer
RUN mkdir -p $COMPOSER_HOME && chown www-data:www-data $COMPOSER_HOME
```



```dockerfile
# syntax=docker/dockerfile:1
# Configure version via --build-arg PHP_IMAGE=8.1-cli This image is for alpine only.

ARG PHP_IMAGE=8
FROM php:${PHP_IMAGE}-alpine as builder

COPY --from=mlocati/php-extension-installer /usr/bin/install-php-extensions /usr/bin/

RUN <<EOF
apk --update add --no-cache --virtual .build-deps $PHPIZE_DEPS
apk --update add --no-cache libressl bash unzip
install-php-extensions apcu intl redis opcache pcov pdo_mysql xdebug zip
apk del .build-deps
rm -rf /usr/local/etc/php/conf.d/docker-php-ext-{pcov,xdebug}.ini
EOF

COPY --link rootfs /

# See https://getcomposer.org/doc/03-cli.md#composer-home
ENV COMPOSER_HOME=/tmp/composer
RUN mkdir -p $COMPOSER_HOME && chown www-data:www-data $COMPOSER_HOME
```

```dockerfile
# syntax=docker/dockerfile:1
# Configure version via --build-arg PHP_IMAGE=8.1-cli This image is for alpine only.

ARG PHP_IMAGE=8
FROM php:${PHP_IMAGE}-alpine as builder

COPY --from=mlocati/php-extension-installer /usr/bin/install-php-extensions /usr/bin/

RUN <<EOF
apk --update add --no-cache --virtual .build-deps $PHPIZE_DEPS
apk --update add --no-cache libressl bash unzip
install-php-extensions apcu intl redis opcache pcov pdo_mysql xdebug zip
apk del .build-deps
rm -rf /usr/local/etc/php/conf.d/docker-php-ext-{pcov,xdebug}.ini
EOF

COPY --link rootfs /

# See https://getcomposer.org/doc/03-cli.md#composer-home
ENV COMPOSER_HOME=/tmp/composer
RUN mkdir -p $COMPOSER_HOME && chown www-data:www-data $COMPOSER_HOME
```

```
# syntax=docker/dockerfile:1
# Configure version via --build-arg PHP_IMAGE=8.1-cli This image is for alpine only.

ARG PHP_IMAGE=8
FROM php:${PHP_IMAGE}-alpine as builder

COPY --from=mlocati/php-extension-installer /usr/bin/install-php-extensions /usr/bin/

RUN <<EOF
apk --update add --no-cache --virtual .build-deps $PHPIZE_DEPS
apk --update add --no-cache libressl bash unzip
install-php-extensions apcu intl redis opcache pcov pdo_mysql xdebug zip
apk del .build-deps
rm -rf /usr/local/etc/php/conf.d/docker-php-ext-{pcov,xdebug}.ini
EOF

COPY --link rootfs /

# See https://getcomposer.org/doc/03-cli.md#composer-home
ENV COMPOSER_HOME=/tmp/composer
RUN mkdir -p $COPOSER_HOME && chown www-data:www-data $COMPOSER_HOME
```

```dockerfile
# syntax=docker/dockerfile:1
# Configure version via --build-arg PHP_IMAGE=8.1-cli This image is for alpine only.

ARG PHP_IMAGE=8
FROM php:${PHP_IMAGE}-alpine as builder

COPY --from=mlocati/php-extension-installer /usr/bin/install-php-extensions /usr/bin/

RUN <<EOF
apk --update add --no-cache --virtual .build-deps $PHPIZE_DEPS
apk --update add --no-cache libressl bash unzip
install-php-extensions apcu intl redis opcache pcov pdo_mysql xdebug zip
apk del .build-deps
rm -rf /usr/local/etc/php/conf.d/docker-php-ext-{pcov,xdebug}.ini
EOF

COPY --link rootfs /

# See https://getcomposer.org/doc/03-cli.md#composer-home
ENV COMPOSER_HOME=/tmp/composer
RUN mkdir -p $COMPOSER_HOME && chown www-data:www-data $COMPOSER_HOME
```

```dockerfile
# syntax=docker/dockerfile:1
# Configure version via --build-arg PHP_IMAGE=8.1-cli This image is for alpine only.

ARG PHP_IMAGE=8
FROM php:${PHP_IMAGE}-alpine as builder

COPY --from=mlocati/php-extension-installer /usr/bin/install-php-extensions /usr/bin/

RUN <<EOF
apk --update add --no-cache --virtual .build-deps $PHPIZE_DEPS
apk --update add --no-cache libressl bash unzip
install-php-extensions apcu intl redis opcache pcov pdo_mysql xdebug zip
apk del .build-deps
rm -rf /usr/local/etc/php/conf.d/docker-php-ext-{pcov,xdebug}.ini
EOF

COPY --link rootfs /

# See https://getcomposer.org/doc/03-cli.md#composer-home
ENV COMPOSER_HOME=/tmp/composer
RUN mkdir -p $COMPOSER_HOME && chown www-data:www-data $COMPOSER_HOME
```

Base PHP image per company

```
docker build image -t example.com/php/8.1-alpine:1.0.0 --build-arg PHP_IMAGE=8.1

docker image push example.com/php/8.1-alpine:1.0.0

FROM example.com/php/8.1-alpine:1.0.0
```



Base PHP image per company

```
docker build image -t example.com/php/8.1-alpine:1.0.0 --build-arg PHP_IMAGE=8.1
```

```
docker image push example.com/php/8.1-alpine:1.0.0
```

```
FROM example.com/php/8.1-alpine:1.0.0
```



Base PHP image per company

```
docker build image -t example.com/php/8.1-alpine:1.0.0 --build-arg PHP_IMAGE=8.1

docker image push example.com/php/8.1-alpine:1.0.0

FROM example.com/php/8.1-alpine:1.0.0
```



Base PHP image per company

```
docker build image -t example.com/php/8.1-alpine:1.0.0 --build-arg PHP_IMAGE=8.1

docker image push example.com/php/8.1-alpine:1.0.0

FROM example.com/php/8.1-alpine:1.0.0
```



One application = one Dockerfile

```
# syntax = docker/dockerfile:1
FROM example.com/php/8.1-alpine:1.0.0 as base
WORKDIR /app

FROM base as dev
RUN \
    --mount=type=cache,id=apk-8.1,target=/var/cache/apk \
    ln -vs /var/cache/apk /etc/apk/cache && \
    apk --update add openssh-client git
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

FROM dev as builder
ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=ssh \
    --mount=type=cache,id=cache,target=$COMPOSER_HOME/cache \
    --mount=type=secret,id=auth.json,target=$COMPOSER_HOME/auth.json,required \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/

FROM base as production
USER www-data
COPY --from=builder --chown=www-data:www-data /build /app
```



```dockerfile
# syntax = docker/dockerfile:1
FROM example.com/php/8.1-alpine:1.0.0 as base
WORKDIR /app

FROM base as dev
RUN \
    --mount=type=cache,id=apk-8.1,target=/var/cache/apk \
    ln -vs /var/cache/apk /etc/apk/cache && \
    apk --update add openssh-client git
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

FROM dev as builder
ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=ssh \
    --mount=type=cache,id=cache,target=$COMPOSER_HOME/cache \
    --mount=type=secret,id=auth.json,target=$COMPOSER_HOME/auth.json,required \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/

FROM base as production
USER www-data
COPY --from=builder --chown=www-data:www-data /build /app
```

```dockerfile
# syntax = docker/dockerfile:1
FROM example.com/php/8.1-alpine:1.0.0 as base
WORKDIR /app

FROM base as dev
RUN \
    --mount=type=cache,id=apk-8.1,target=/var/cache/apk \
    ln -vs /var/cache/apk /etc/apk/cache && \
    apk --update add openssh-client git
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

FROM dev as builder
ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=ssh \
    --mount=type=cache,id=cache,target=$COMPOSER_HOME/cache \
    --mount=type=secret,id=auth.json,target=$COMPOSER_HOME/auth.json,required \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/

FROM base as production
USER www-data
COPY --from=builder --chown=www-data:www-data /build /app
```

PHP Russia
2022

```dockerfile
# syntax = docker/dockerfile:1
FROM example.com/php/8.1-alpine:1.0.0 as base
WORKDIR /app

FROM base as dev
RUN \
    --mount=type=cache,id=apk-8.1,target=/var/cache/apk \
    ln -vs /var/cache/apk /etc/apk/cache && \
    apk --update add openssh-client git
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

FROM dev as builder
ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=ssh \
    --mount=type=cache,id=cache,target=$COMPOSER_HOME/cache \
    --mount=type=secret,id=auth.json,target=$COMPOSER_HOME/auth.json,required \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/

FROM base as production
USER www-data
COPY --from=builder --chown=www-data:www-data /build /app
```

```dockerfile
# syntax = docker/dockerfile:1
FROM example.com/php/8.1-alpine:1.0.0 as base
WORKDIR /app

FROM base as dev
RUN \
    --mount=type=cache,id=apk-8.1,target=/var/cache/apk \
    ln -vs /var/cache/apk /etc/apk/cache && \
    apk --update add openssh-client git
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

FROM dev as builder
ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=ssh \
    --mount=type=cache,id=cache,target=$COMPOSER_HOME/cache \
    --mount=type=secret,id=auth.json,target=$COMPOSER_HOME/auth.json,required \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/

FROM base as production
USER www-data
COPY --from=builder --chown=www-data:www-data /build /app
```

```
# syntax = docker/dockerfile:1
FROM example.com/php/8.1-alpine:1.0.0 as base
WORKDIR /app

FROM base as dev
RUN \
    --mount=type=cache,id=apk-8.1,target=/var/cache/apk \
    ln -vs /var/cache/apk /etc/apk/cache && \
    apk --update add openssh-client git
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

FROM dev as builder
ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=ssh \
    --mount=type=cache,id=cache,target=$COMPOSER_HOME/cache \
    --mount=type=secret,id=auth.json,target=$COMPOSER_HOME/auth.json,required \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/

FROM base as production
USER www-data
COPY --from=builder --chown=www-data:www-data /build /app
```

```dockerfile
# syntax = docker/dockerfile:1
FROM example.com/php/8.1-alpine:1.0.0 as base
WORKDIR /app

FROM base as dev
RUN \
    --mount=type=cache,id=apk-8.1,target=/var/cache/apk \
    ln -vs /var/cache/apk /etc/apk/cache && \
    apk --update add openssh-client git
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

FROM dev as builder
ENV COMPOSER_HOME=/tmp/composer
RUN \
    --mount=type=bind,target=/app,rw \
    --mount=type=ssh \
    --mount=type=cache,id=cache,target=$COMPOSER_HOME/cache \
    --mount=type=secret,id=auth.json,target=$COMPOSER_HOME/auth.json,required \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/

FROM base as production
USER www-data
COPY --from=builder --chown=www-data:www-data /build /app
```



The fastest composer install

```
RUN \
    --mount=type=bind,target=/app,rw \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



The fastest composer install

```
RUN \
    --mount=type=bind,target=/app,rw \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



The fastest composer install

```
RUN \
    --mount=type=bind,target=/app,rw \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



The fastest composer install

```
RUN \
    --mount=type=bind,target=/app,rw \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```





The fastest composer install

```
RUN \
    --mount=type=bind,target=/app,rw \
    composer install --no-dev --classmap-authoritative && \
    cp -r /app/ /build/
```



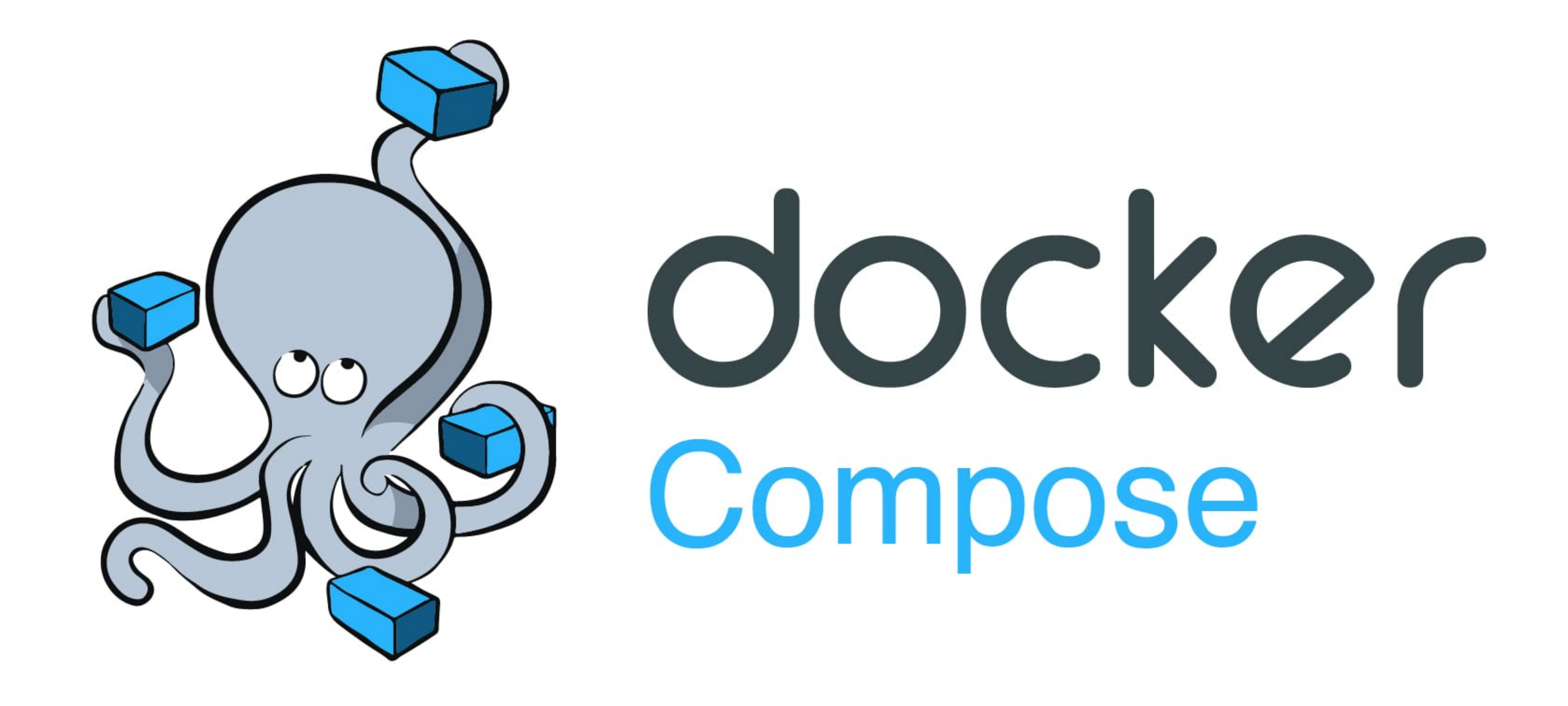
Running entire PHP stack via «docker compose»











Compose is an orchestration tool that makes spinning up multi-container distributed applications with Docker an effortless task.



Base docker-compose.yml

```yaml
services:
  application.docker:
    image: nginx:stable-alpine
    volumes:
      - ./docker/nginx/sites.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
    working_dir: /app
    user: "${CURRENT_USER:-0}"
    environment:
      COMPOSER_HOME: /tmp/composer
    volumes:
      - ./:/app:rw
      - ${HOME}/.composer:/tmp/composer
      - ./docker/php-fpm/php.ini:/usr/local/etc/php/conf.d/php.ini
```



```yaml
services:
  application.docker:
    image: nginx:stable-alpine
    volumes:
      - ./docker/nginx/sites.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
    working_dir: /app
    user: "${CURRENT_USER:-0}"
    environment:
      COMPOSER_HOME: /tmp/composer
    volumes:
      - ./:/app:rw
      - ${HOME}/.composer:/tmp/composer
      - ./docker/php-fpm/php.ini:/usr/local/etc/php/conf.d/php.ini
```



```yaml
services:
  application.docker:
    image: nginx:stable-alpine
    volumes:
      - ./docker/nginx/sites.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
    working_dir: /app
    user: "${CURRENT_USER:-0}"
    environment:
      COMPOSER_HOME: /tmp/composer
    volumes:
      - ./:/app:rw
      - ${HOME}/.composer:/tmp/composer
      - ./docker/php-fpm/php.ini:/usr/local/etc/php/conf.d/php.ini
```



```yaml
services:
  application.docker:
    image: nginx:stable-alpine
    volumes:
      - ./docker/nginx/sites.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
    working_dir: /app
    user: "${CURRENT_USER:-0}"
    environment:
      COMPOSER_HOME: /tmp/composer
    volumes:
      - ./:/app:rw
      - ${HOME}/.composer:/tmp/composer
      - ./docker/php-fpm/php.ini:/usr/local/etc/php/conf.d/php.ini
```



```yaml
services:
  application.docker:
    image: nginx:stab
    volumes:
      - ./docker/ngin
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
    working_dir: /app
    user: "${CURRENT_USER:-0}"
    environment:
      COMPOSER_HOME: /tmp/composer
    volumes:
      - ./:/app:rw
      - ${HOME}/.composer:/tmp/composer
      - ./docker/php-fpm/php.ini:/usr/local/etc/php/conf.d/php.ini
```

```dockerfile
# syntax = docker/dockerfile:1
FROM example.com/php/8.1-alpine:1.0.0 as base
WORKDIR /app

FROM base as dev
RUN \
    --mount=type=cache,id=apk-8.1,target=/var/cache/apk \
    ln -vs /var/cache/apk /etc/apk/cache && \
    apk --update add openssh-client git
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer
```

```yaml
services:
  application.docker:
    image: nginx:stable-alpine
    volumes:
      - ./docker/nginx/sites.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
    working_dir: /app
    user: "${CURRENT_USER:-0}"
    environment:
      COMPOSER_HOME: /tmp/composer
    volumes:
      - ./:/app:rw
      - ${HOME}/.composer:/tmp/composer
      - ./docker/php-fpm/php.ini:/usr/local/etc/php/conf.d/php.ini
```



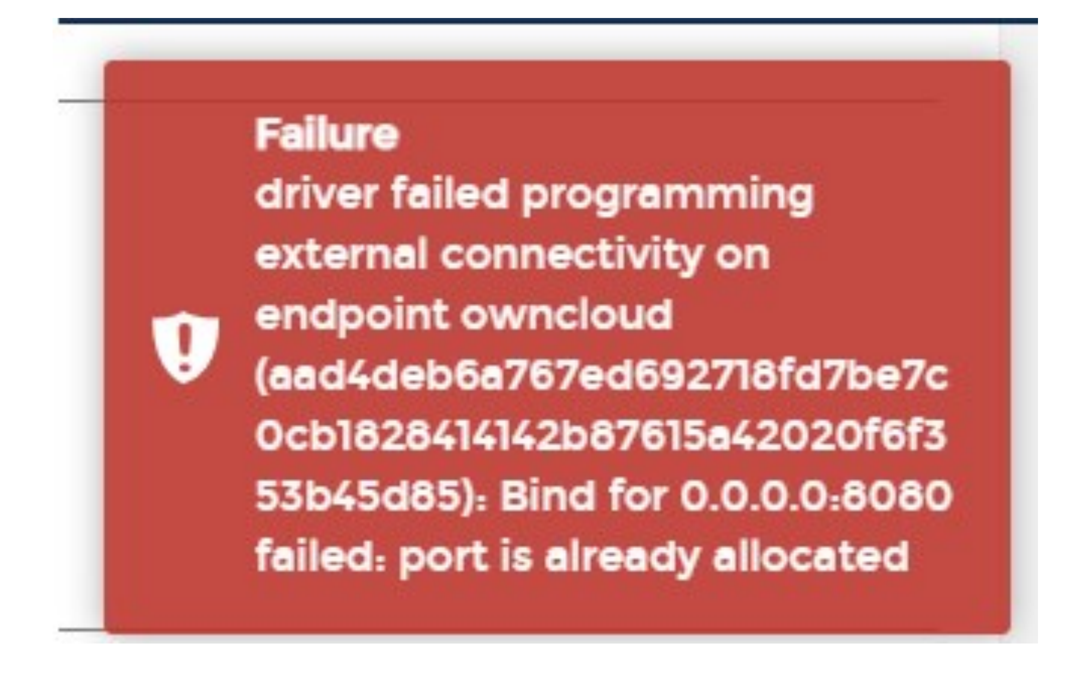
```yaml
services:
  application.docker:
    image: nginx:stable-alpine
    volumes:
      - ./docker/nginx/sites.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
    working_dir: /app
    user: "${CURRENT_USER:-0}"
    environment:
      COMPOSER_HOME: /tmp/composer
    volumes:
      - ./:/app:rw
      - ${HOME}/.composer:/tmp/composer
      - ./docker/php-fpm/php.ini:/usr/local/etc/php/conf.d/php.ini
```

```yaml
services:
  application.docker:
    image: nginx:stable-alpine
    volumes:
      - ./docker/nginx/sites.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
    working_dir: /app
    user: "${CURRENT_USER:-0}"
    environment:
      COMPOSER_HOME: /tmp/composer
    volumes:
      - ./:/app:rw
      - ${HOME}/.composer:/tmp/composer
      - ./docker/php-fpm/php.ini:/usr/local/etc/php/conf.d/php.ini
```

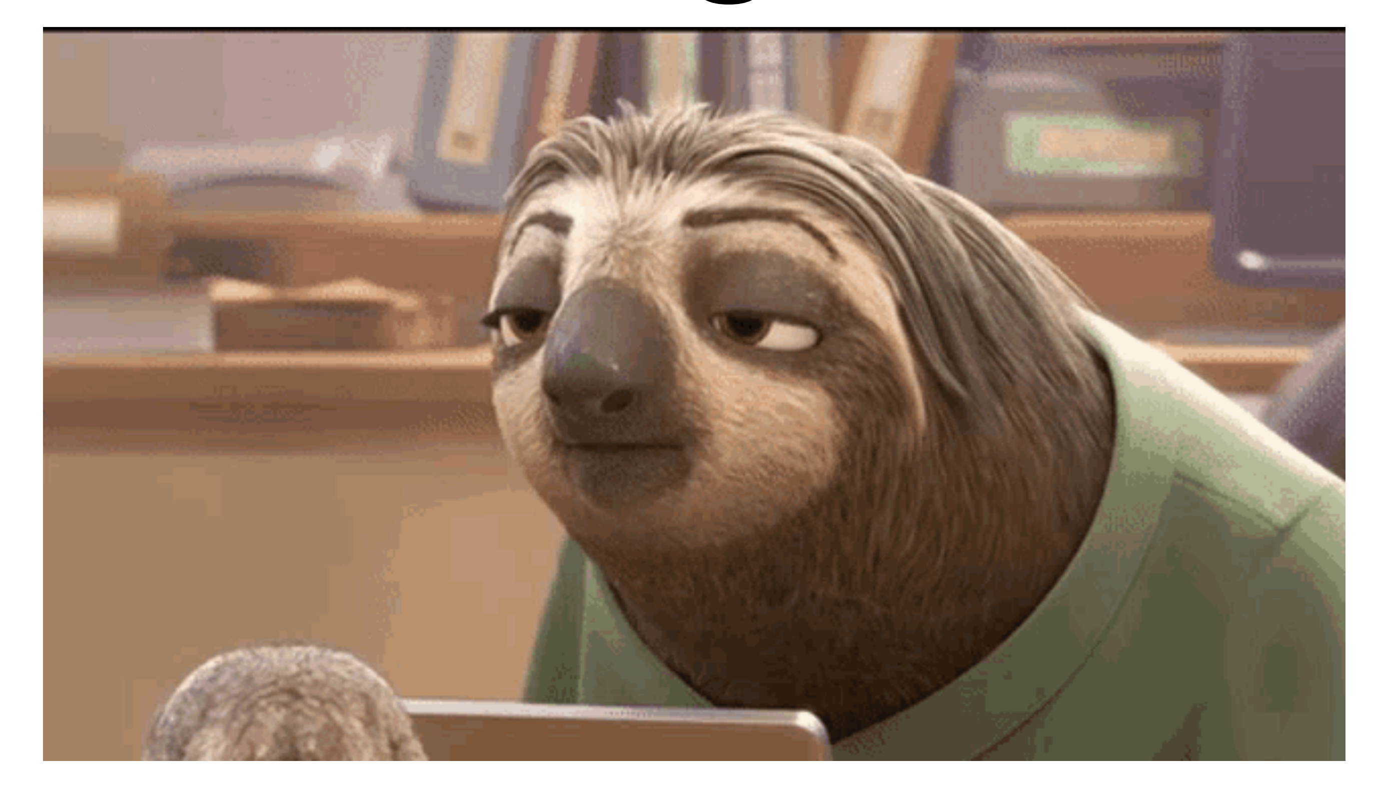
Problem with docker compose



Do not use «ports» section for CI/CD pipeline, as you will have bind troubles due to the occupied port on build machines.



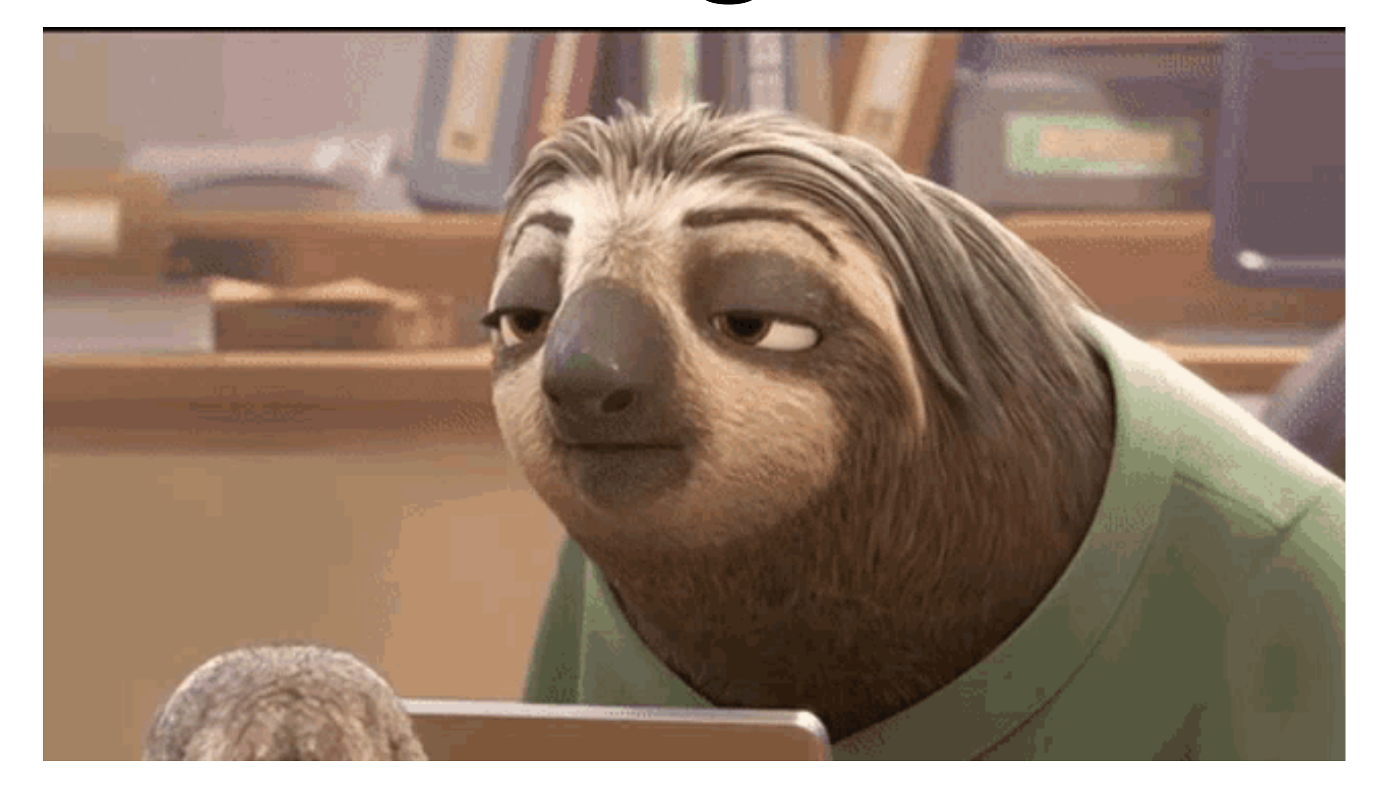
Slow code-coverage collection



Xdebug can be very slow for code coverage, even version 3, better to use pcov extension for test env.



Slow code-coverage collection



Xdebug can be very slow for code coverage, even version 3, better to use pcov extension for test env.



Ok Fix !!!



Idea: use definition merging for docker-compose.

«...Using multiple Compose files enables you to customize a Compose application for different environments or different workflows...»

Dev settings: put to the **docker-compose.override.yml.dist**

Test settings: put to the **docker-compose.test.yml**

Use Makefile to do conditional include of proper files



docker-compose.override.yml.dist

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```



```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

ISSIA

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

```yaml
services:
  application.docker:
    image: nginx:stable-alpine
    volumes:
      - ./docker/nginx/sites.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    build:
      context: .
      target: dev
      dockerfile: ./docker/php-fpm/Dockerfile
```

issia

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

issia

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

ıssia

```yaml
services:
  application.docker:
    networks:
      - your_company
      - default
    ports:
      - 127.0.0.1:${HOST_PORT:-80}:80

  api:
    build:
      context: docker/php-fpm
      dockerfile: Dockerfile
    networks:
      - your_company
      - default
    env_file: .env
    volumes:
      - ./docker/php-fpm/ext-xdebug.ini:/usr/local/etc/php/conf.d/ext-xdebug.ini

networks:
  your_company:
    external: true
```

docker-compose.test.yml

```yaml
services:
  api:
    env_file: .env.test
    volumes:
      - ./docker/php-fpm/ext-pcov.ini:/usr/local/etc/php/conf.d/ext-pcov.ini

wiremock:
  image: "wiremock/wiremock:2.34.0-alpine"
  networks:
    default:
      aliases:
        - 'payment-provider.wiremock'
        - 'google-search.wiremock'
```



docker-compose.test.yml

```yaml
services:
  api:
    env_file: .env.test
    volumes:
      - ./docker/php-fpm/ext-pcov.ini:/usr/local/etc/php/conf.d/ext-pcov.ini

wiremock:
  image: "wiremock/wiremock:2.34.0-alpine"
  networks:
    default:
      aliases:
        - 'payment-provider.wiremock'
        - 'google-search.wiremock'
```



docker-compose.test.yml

```yaml
services:
  api:
    env_file: .env.test
    volumes:
      - ./docker/php-fpm/ext-pcov.ini:/usr/local/etc/php/conf.d/ext-pcov.ini

wiremock:
  image: "wiremock/wiremock:2.34.0-alpine"
  networks:
    default:
      aliases:
        - 'payment-provider.wiremock'
        - 'google-search.wiremock'
```

docker-compose.test.yml

```yaml
services:
  api:
    env file: .env.test
    volumes:
      - ./docker/php-fpm/ext-pcov.ini:/usr/local/etc/php/conf.d/ext-pcov.ini

wiremock:
  image: "wiremock/wiremock:2.34.0-alpine"
  networks:
    default:
      aliases:
        - 'payment-provider.wiremock'
        - 'google-search.wiremock'
```



docker-compose.test.yml

```yaml
services:
  api:
    env_file: .env.test
    volumes:
      - ./docker/php-fpm/ext-pcov.ini:/usr/local/etc/php/conf.d/ext-pcov.ini

wiremock:
  image: "wiremock/wiremock:2.34.0-alpine"
  networks:
    default:
      aliases:
        - 'payment-provider.wiremock'
        - 'google-search.wiremock'
```



12-factor environment injection

```
#.env.dist
COMPOSE_PROJECT_NAME=phprussia22-local
HOST_PORT=8080

CONFIG_MYSQL_HOST=mysql-shared.example.com
CONFIG_MYSQL_USER=my-service
CONFIG_MYSQL_PASS=*****

#.env.test
CONFIG_MYSQL_HOST=mysql
CONFIG_MYSQL_USER=root
CONFIG_MYSQL_PASS=root
```



12-factor environment injection

```
#.env.dist
COMPOSE_PROJECT_NAME=phprussia22-local
HOST_PORT=8080

CONFIG_MYSQL_HOST=mysql-shared.example.com
CONFIG_MYSQL_USER=my-service
CONFIG_MYSQL_PASS=*****

#.env.test
CONFIG_MYSQL_HOST=mysql
CONFIG_MYSQL_USER=root
CONFIG_MYSQL_PASS=root
```



12-factor environment injection

```
#.env.dist
COMPOSE_PROJECT_NAME=phprussia22-local
HOST_PORT=8080

CONFIG_MYSQL_HOST=mysql-shared.example.com
CONFIG_MYSQL_USER=my-service
CONFIG_MYSQL_PASS=*****

#.env.test
CONFIG_MYSQL_HOST=mysql
CONFIG_MYSQL_USER=root
CONFIG_MYSQL_PASS=root
```



12-factor environment injection

```
#.env.dist
COMPOSE_PROJECT_NAME=phprussia22-local
HOST_PORT=8080

CONFIG_MYSQL_HOST=mysql-shared.example.com
CONFIG_MYSQL_USER=my-service
CONFIG_MYSQL_PASS=*****

#.env.test
CONFIG_MYSQL_HOST=mysql
CONFIG_MYSQL_USER=root
CONFIG_MYSQL_PASS=root
```



Isolating each env from another

COMPOSE_PROJECT_NAME

Sets the project name. This value is prepended along with the service name to the container's name on startup.

For example, if your project name is `myapp` and it includes two services `db` and `web`, then Compose starts containers named `myapp-db-1` and `myapp-web-1` respectively.

- **Defaults to:** the `basename` of the project directory.



Isolating each env from another

COMPOSE_PROJECT_NAME

Sets the project name. This value is prepended along with the service name to the container's name on startup.

For example, if your project name is `myapp` and it includes two services `db` and `web`, then Compose starts containers named `myapp-db-1` and `myapp-web-1` respectively.

- **Defaults to:** the `basename` of the project directory.



Isolating each env from another

COMPOSE_PROJECT_NAME

Sets the project name. This value is prepended along with the service name to the container's name on startup.

For example, if your project name is `myapp` and it includes two services `db` and `web`, then Compose starts containers named `myapp-db-1` and `myapp-web-1` respectively.

- **Defaults to:** the `basename` of the project directory.

So, COMPOSE_PROJECT_NAME stored **in the relevant .env** file (eg. .env.dist) helps us to isolate services by env in one folder.



Combining several definitions for env

```
# Makefile.local
DOCKER_COMPOSE=docker compose

# Makefile.test
DOCKER_COMPOSE=docker compose --env-file .env.test -f docker-
compose.yml -f docker-compose.test.yml

# Makefile
env ?= local
include Makefile.${env}

start: ## Start environment
	@echo 'Starting Docker containers'
	$(DOCKER_COMPOSE) up -d --build --remove-orphans
```



Combining several definitions for env

```
# Makefile.local
DOCKER_COMPOSE=docker compose

# Makefile.test
DOCKER_COMPOSE=docker compose --env-file .env.test -f docker-
compose.yml -f docker-compose.test.yml

# Makefile
env ?= local
include Makefile.${env}

start: ## Start environment
	@echo 'Starting Docker containers'
	$(DOCKER_COMPOSE) up -d --build --remove-orphans
```



Combining several definitions for env

```
# Makefile.local
DOCKER_COMPOSE=docker compose

# Makefile.test
DOCKER_COMPOSE=docker compose --env-file .env.test -f docker-
compose.yml -f docker-compose.test.yml

# Makefile
env ?= local
include Makefile.${env}

start: ## Start environment
	@echo 'Starting Docker containers'
	$(DOCKER_COMPOSE) up -d --build --remove-orphans
```



Combining several definitions for env

```
# Makefile.local
DOCKER_COMPOSE=docker compose

# Makefile.test
DOCKER_COMPOSE=docker compose --env-file .env.test -f docker-
compose.yml -f docker-compose.test.yml

# Makefile
env ?= local
include Makefile.${env}

start: ## Start environment
	@echo 'Starting Docker containers'
	$(DOCKER_COMPOSE) up -d --build --remove-orphans
```



Combining several definitions for env

```
# Makefile.local
DOCKER_COMPOSE=docker compose

# Makefile.test
DOCKER_COMPOSE=docker compose --env-file .env.test -f docker-
compose.yml -f docker-compose.test.yml

# Makefile
env ?= local
include Makefile.${env}

start: ## Start environment
	@echo 'Starting Docker containers'
	$(DOCKER_COMPOSE) up -d --build --remove-orphans
```



Running PHP inside Kubernetes



kubernetes

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.




Blue elePHPant, flying in clouds from computers, cyberpunk
Midjourney(c)

Ingredients

- Support of 12-factor manifest PHP by application
- Docker image of our PHP application
- CI/CD Pipeline definition to create a docker image from source code, triggered by push
- Registry for pushing Docker images and Helm charts
- **Defined Kubernetes resources for app or Helm chart.**
- K8s-ready platform (on-prem or any cloud, eg AWS/GCP)



The Twelve-Factor app manifest

- Use declarative formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a clean contract with the underlying operating system, offering maximum portability between execution environments;
- Are suitable for deployment on modern cloud platforms, obviating the need for servers and systems administration;
- Minimize divergence between development and production, enabling continuous deployment for maximum agility;
- And can scale up without significant changes to tooling, architecture, or development practices.

https://12factor.net/



Kubernetes service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-application
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: my-application
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
```



Kubernetes service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-application
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: my-application
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
```



Kubernetes service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-application
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: my-application
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
```



Kubernetes service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-application
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: my-application
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
```



Kubernetes deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  selector:
    matchLabels:
      app: my-application
  template:
    metadata:
      labels:
        app: my-application
    spec:
      containers:
        - name: nginx
          image: nginx:stable-alpine
          ports:
            - containerPort: 80
          volumeMounts:
            - name: my-nginx-configmap
              mountPath: "/etc/nginx/conf.d/"
              readOnly: true
        - name: php-fpm
          image: {{ container.image }}
          ports:
            - containerPort: 9000
```



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  selector:
    matchLabels:
      app: my-application
  template:
    metadata:
      labels:
        app: my-application
    spec:
      containers:
        - name: nginx
          image: nginx:stable-alpine
          ports:
            - containerPort: 80
          volumeMounts:
            - name: my-nginx-configmap
              mountPath: "/etc/nginx/conf.d/"
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  selector:
    matchLabels:
      app: my-application
  template:
    metadata:
      labels:
        app: my-application
    spec:
      containers:
        - name: nginx
          image: nginx:stable-alpine
          ports:
            - containerPort: 80
          volumeMounts:
            - name: my-nginx-configmap
              mountPath: "/etc/nginx/conf.d/"
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  selector:
    matchLabels:
      app: my-application
  template:
    metadata:
      labels:
        app: my-application
    spec:
      containers:
        - name: nginx
          image: nginx:stable-alpine
          ports:
            - containerPort: 80
          volumeMounts:
            - name: my-nginx-configmap
              mountPath: "/etc/nginx/conf.d/"
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  selector:
    matchLabels:
      app: my-application
  template:
    metadata:
      labels:
        app: my-application
    spec:
      containers:
        - name: nginx
          image: nginx:stable-alpine
          ports:
            - containerPort: 80
          volumeMounts:
            - name: my-nginx-configmap
              mountPath: "/etc/nginx/conf.d/"
```

```yaml
metadata:
  labels:
    app: my-application
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
      - containerPort: 80
    volumeMounts:
      - name: my-nginx-configmap
        mountPath: "/etc/nginx/conf.d/"
        readOnly: true
  - name: php-fpm
    image: {{ container.image }}
    ports:
      - containerPort: 9000
    envFrom:
      - secretRef:
          name: my-php-fpm-secret
```

```yaml
metadata:
  labels:
    app: my-application
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
      volumeMounts:
        - name: my-nginx-configmap
          mountPath: "/etc/nginx/conf.d/"
          readOnly: true
    - name: php-fpm
      image: {{ container.image }}
      ports:
        - containerPort: 9000
      envFrom:
        - secretRef:
            name: my-php-fpm-secret
```

Nginx configmap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-nginx-configmap
data:
  default.conf: |
    server {
      listen      80 default_server;
      server_name _;
      root        /app/web;
      index       index.php;
      sendfile    off;
      location / {
        try_files $uri @rewriteapp;
      }
      location @rewriteapp {
        rewrite ^(.*)$ /index.php/$1 last;
      }
      location ~ ^/.+\.php(/|$) {
        fastcgi_pass  127.0.0.1:9000;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include       fastcgi_params;
      }
    }
```



Nginx configmap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-nginx-configmap
data:
  default.conf: |
    server {
      listen       80 default_server;
      server_name _;
      root         /app/web;
      index        index.php;
      sendfile     off;
      location / {
        try_files $uri @rewriteapp;
      }
      location @rewriteapp {
        rewrite ^(.*)$ /index.php/$1 last;
      }
      location ~ ^/.+\.php(/|$) {
        fastcgi_pass  127.0.0.1:9000;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include       fastcgi_params;
      }
    }
```



Nginx configmap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-nginx-configmap
data:
  default.conf: |
    server {
      listen       80 default_server;
      server_name _;
      root         /app/web;
      index        index.php;
      sendfile     off;
      location / {
        try_files $uri @rewriteapp;
      }
      location @rewriteapp {
        rewrite ^(.*)$ /index.php/$1 last;
      }
      location ~ ^/.+\.php(/|$) {
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include        fastcgi_params;
      }
    }
```



All good, but...



Docker Deployment kubectl run nginx --image=nginx
Pods

Ingress ConfigMap
Job Service
Secret

CRD PersistentVolumeClaim Affinity StatefulSet
Helm templating Dashboard HPA
Log management Init Containers Tolerations ResourceLimit

LoadBalancing PID namespace sharing Hard and Soft Evictions
Prometheus Grafana Healthchecks PromQL
Evictions API (PDB) Terraform Service Discovery
QoS (cgroups, oom_score_adj) HPA/VPA based on custom metrics

Egress gateway Apparmor Enterprise-level security PodSecurityPolicy Customizable monitoring for all Kubernetes objects
OpenID Connect Dynamic Admission Control Audit logging Dynamic StorageClass provisioning
Image validation and signing Resource sharing seccomp NetworkPolicy cluster-autoscaler
Cert-manager Mutating and Admission webhooks Ingress Monitoring descheduler Longterm Prometheus
Dex OpenPolicyAgent Prometheus Query caching Ingress autoscaling Certificate renewal
Advanced control plane configuration IaC for Grafana (dashboards, datasources) Multitenancy RBAC
Kubernetes Upgrade etcd cluster management Custom Resources validation and conversion

Monitoring underlying infrastructure Terraform-managed infrastructure Cost analysis (cloud provider resources)
Network requests tracing CNI (Cilium, calico, flannel, integration with cloud provider VPCs) sysctl
Control plane maintenance windows Service Mesh (Istio) Highly available control plane components
Certificate management in k8s control plane node-local-dns Changes monitoring
Operators Configuring kubectl for Remote Access Non-destructive update applier

Node's OS configuration Spot instances gvisor Cloud Provider Integration
security sandboxing Node Rolling update CSI Cluster API
kata containers Managing Instance Groups OS level Node bootstrap
Chaos testing/engineering CRI Advanced scheduling MetalLB keepalived

Patching control-plane-components KEPs Directly interacting with etcd

FLANT




Deployment
Docker
kubectl run nginx --image=nginx
Pods
Ingress
ConfigMap
Job
Service
Secret
CRD
PersistentVolumeClaim
Affinity
StatefulSet
Helm templating
Dashboard
HPA
Log management
Init Containers
Tolerations
ResourceLimit


CRD
PersistentVolumeClaim
Affinity
StatefulSet
Helm templating
Dashboard
HPA
Log management
Init Containers
Tolerations
ResourceLimit
LoadBalancing
PID namespace sharing
Hard and Soft Evictions
Prometheus
Grafana
Healthchecks
PromQL
Evictions API (PDB)
Terraform
Service Discovery
QoS (cgroups, oom_score_adj)
HPA/VPA based on custom metrics
Egress gateway
Apparmor
Enterprise-level security
PodSecurityPolicy
Customizable monitoring for all Kubernetes objects
OpenID Connect
Dynamic Admission Control
Audit logging
Dynamic StorageClass provisioning
Image validation and signing
Resource sharing
seccomp
NetworkPolicy
cluster-autoscaler
Cert-manager
Mutating and Admission webhooks
Ingress Monitoring
descheduler
Longterm Prometheus
Dex
OpenPolicyAgent
Prometheus Query caching
Ingress autoscaling
Certificate renewal
Advanced control plane configuration
IaC for Grafana (dashboards, datasources)
Multitenancy
RBAC



Egress gateway
Apparmor
Enterprise-level security
PodSecurityPolicy
Customizable monitoring for all Kubernetes objects
OpenID Connect
Dynamic Admission Control
Audit logging
Dynamic StorageClass provisioning
Image validation and signing
Resource sharing
seccomp
NetworkPolicy
cluster-autoscaler
Cert-manager
Mutating and Admission webhooks
Ingress Monitoring
descheduler
Longterm Prometheus
Dex
OpenPolicyAgent
Prometheus Query caching
Ingress autoscaling
Certificate renewal
Advanced control plane configuration
IaC for Grafana (dashboards, datasources)
Multitenancy
RBAC
Kubernetes Upgrade
etcd cluster management
Custom Resources validation and conversion

Monitoring underlying infrastructure
Terraform-managed infrastructure
Cost analysis (cloud provider resources)
Network requests tracing
CNI (Cilium, calico, flannel, integration with cloud provider VPCs)
sysctl
Control plane maintenance windows
Service Mesh (Istio)
Highly available control plane components
Certificate management in k8s control plane
node-local-dns
Changes monitoring

Kubernetes Upgrade
etcd cluster management
Custom Resources validation and conversion

Monitoring underlying infrastructure
Terraform-managed infrastructure
Cost analysis (cloud provider resources)
Network requests tracing
CNI (Cilium, calico, flannel, integration with cloud provider VPCs)
sysctl
Control plane maintenance windows
Service Mesh (Istio)
Highly available control plane components
Certificate management in k8s control plane
node-local-dns
Changes monitoring
Operators
Configuring kubectl for Remote Access
Non-destructive update applier

Node's OS configuration
Spot instances
gvisor
Cloud Provider Integration
security sandboxing
Node Rolling update
CSI
Cluster API
kata containers
Managing Instance Groups
OS level Node bootstrap
Chaos testing/engineering
CRI
Advanced scheduling
MetalLB
keepalived

Patching control-plane-components
KEPs
Directly interacting with etcd



Looks like it is impossible to do K8s in 30 minutes...





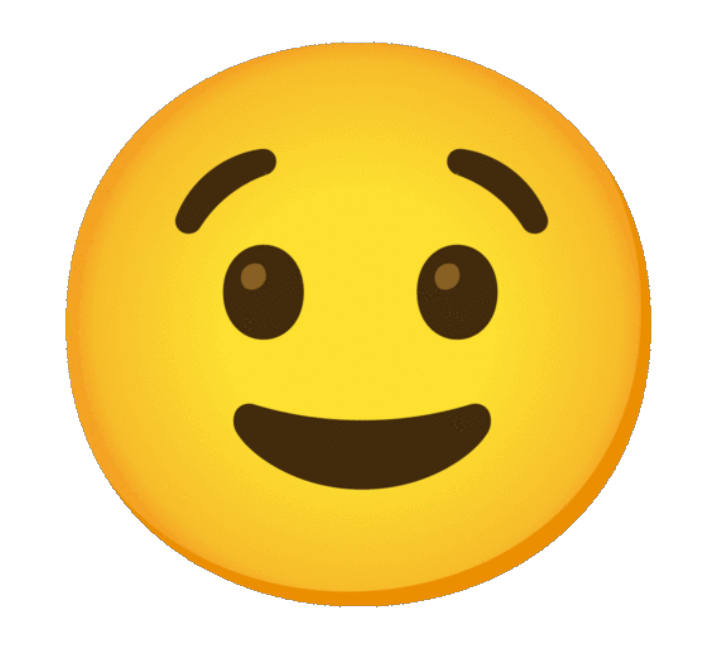
Looks like it is impossible to do K8s in 30 minutes...





But you can try to ask me after my talk!

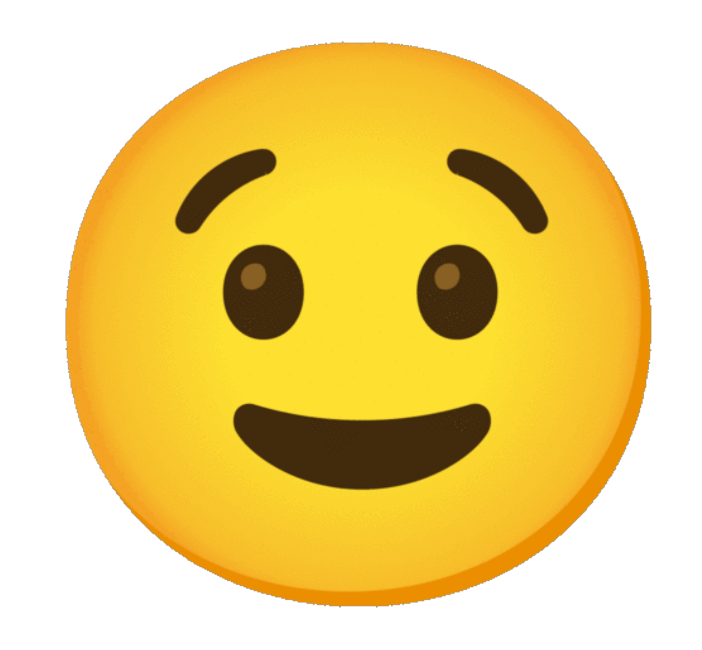
That's all! Thank you!





But you can try to ask me after my talk!

That's all! Thank you!





Обратная связь
и комментарии по
докладу по ссылке



